

Q133 : 克隆图

1. 前序递归，进行结点拷贝即可。
2. 层次遍历，用map：记录新旧结点的映射，来标记一个点是否被复制过。

Q207 : 课程表： 拓扑排序 dfs拓扑排序

1. 可以使用常规的拓扑排序，但是说实话，不常写，代码量也偏长。十分推荐 dfs的拓扑排序。

```

public void dfs(int u) {
    visited[u] = 1;
    // 标记结点u正在访问，还没完成全部子节点的访问。
    for (int v: edges.get(u)) {
        // 子节点未访问，则可以访问下去
        if (visited[v] == 0) {
            dfs(v);
            // valid是一个全局变量，若上边dfs把valid改为false，说明
            // tuopu失败。

            if (!valid) {
                return;
            }
        } else if (visited[v] == 1) {
            // ==1,说明遇到了正在访问但是未完成的点，
            valid = false;
            // 这就说明v是父节点，又是子节点，说明遇到了环。
            return;
            // 修改为false，并返回。
        }
    }
    visited[u] = 2;
    // 当所有子节点完成访问，标记访问完了。
    // 如果需要序列，则进行保存。
    // ans.addFirst(u)
    // 使用头插法加入 u。因为加入结点的顺序恰好和正常顺序逆序。
}

```

Q332 : 重新安排行程： 一笔画问题 飞机票构成一幅图。然后找到一条路径使用完所有机票。输出每次到达的位置。从JFK出发

1. euler问题：通过dfs，把走过的边标记占用，或者直接删除。当一个结点的所有子结点都访问完了，或者说当一个结点的所有边被标记了.那么就输出该点。

```

public void dfs(String curr) {
    while (map.containsKey(curr) && map.get(curr).size() > 0) {
        String tmp = map.get(curr).poll();
    }
}

```

```

        dfs(tmp);
    }
    itinerary.add(curr);
}

```

Q399 : 除法求值 $a \rightarrow b$: 表示 a/b

并查集 $a \rightarrow b \rightarrow c$ $d \rightarrow r \rightarrow c$

1. a, d 分别是 c 的两个子节点。则 $a/d == > a \rightarrow c \rightarrow d == > a \rightarrow c * c \rightarrow d == > a \rightarrow c / d \rightarrow c == a/d$
2. $a/d == a \rightarrow c / d \rightarrow c$, 所以分别求 $a \rightarrow c$, $d \rightarrow c$ 就是分子和分母。
3. 按照这个思想, 我们可以构造并查集, 若一条边分别在两个集合, 就把集合合并。
4. 优化: 路径压缩。 $a \rightarrow c$ 的路径可能很长。我们可以通过路径压缩, 减少找 c 的过程。

```

HashMap<String,String> p; // p[x] 记录节点 x 的父节点
HashMap<String,Double> d; //d[x] 记录节点 x 到父节点的距离 (即 x /
y)
此时d.get(a),就是a/root,d.get(b)就是b/root

```

Q765 : 情侣牵手

并查集 最后 $ans = n - \text{并查集个数}$ 。

1. 这种情况可以用并查集来表示。 假设有 n 对椅子。 从 $0 \sim 2*n-1$ 个椅子。 下标 01 是一对, 下标 23 是一对, 依次类推。
2. 现在坐在上边的人就是 $01, 23, \dots$ 。现在对人进行交换。 $[0, 2, 1, 3]$
3. 如何连接并查集, 对于第一对椅子, 现在坐着 $0, 2$ 这两人不是情侣。 对 $0/2 = 0 \ 2/2 = 1$ 说明这两人一个属于第 0 对, 一个属于第 1 对。 说明第 0 对和第 1 对发生了交换。 把集合 $0, 1$ 合并。 即第 0 对椅子和第 1 对椅子是同一个集合了。

迪杰斯特拉算法模板

Q778 : 水位上升的泳池中游泳 : 在一个 $N \times N$ 的坐标方格 $grid$ 中, 每一个方格的值 $grid[i][j]$ 表示在位置 (i,j) 的平台高度

当时间为 t 时, 此时雨水导致水池中任意位置的水位为 t 。你可以从一个平台游向四周相邻的任意一个平台, 但是前提是此时水位必须同时淹没这两个平台。假定你可以瞬间移动无限距离。你从坐标方格的左上平台 $(0, 0)$ 出发。最少耗时多久你才能到达坐标方格的右下平台 $(N-1, N-1)$?

$0. grid[i][j]$ 是 $[0, \dots, N*N - 1]$ 的排列. 也就是说, **没有重复的高度** 并查集的前提。

1. Dijkstra 算法:

PriorityQueue<int[]> pq int[] : {x,y,d} 点x,y到已经确认集合的最短距离为 d。优先队列按照d排序。每次都取最小的加入到集合中。然后使用该xy点作为集合中一点，更新到他邻点的距离。加入优先队列。

- 二分法：平台的高度最小是0，最大len. 那么ans高度必然在0到len之间。通过二分高度。mid = (left+right)/2,若水的高度是mid。然后从启动到终点通过 层次遍历。看水的高度是mid时，能否到达终点。不能说明mid小了。否则说明大了. ans每次取最小值可以到达的最小值。

- 并查集。模拟水位上升的过程。**没有重复的高度**

```
index[grid[i][j]] = getIndex(i, j);  grid[i][j] = h,
等价于 index[h] = getIndex(i, j)    下标h是高度。 值是 h出现
```

在图中的 索引。

我们枚举高度时， 就可以通过 index[h]，直到水位为h的点在图中什么位置。（高度不重复）

所以 index是 高度：位置 的映射。一个高度恰好可以映射一个坐标。

当水位是h时，我们获取其对应点的位置是 x,y。若水位高于四个邻接点，那么就合并到一个集合。最终当

起点和终点在一个集合的时候，说明水位到达了。

并查集 & 迪杰斯特拉

Q1631 : 最小体力消耗路径： 基本和Q778相同。

- 二分法来写的话，其实是比较好理解的。并查集的转化问题，有时不好处理。
- 由于我们需要找到从左上角到右下角的最短路径，因此我们可以将图中的所有边按照权值从小到大进行排序，并依次加入并查集中。当我们加入一条权值为 x 的边之后，如果左上角和右下角从非连通状态变为连通状态，那么 x 即为答案。
- 最短路径就不写了，能不用就不用。**

并查集

Q839 : 相似字符串组

如果交换字符串 x 中的两个不同位置的字母，使得它和字符串 y 相等，那么称 x 和 y 两个字符串相似

相似，则可以归到一个集合中。如果x,y相似， y,z相似。 那么xyz在同一个集合内。 称为一组。问最后有多少组。

并查集直接遍历连接一遍 即可。

并查集

Q947 移除最多的同行或同列石头： n 块石头放置在二维平面中的一些整数坐标点上。

- 若同行或者同列，有其他石头，那么该石头可以被移除。 问，当前矩阵中，最多可以移除多少个石头。

1. $ans = n - \text{并查集个数}$
2. 我们规定，若两个石头，同行或者同列，那么两颗石头属于同一个集合。
3. 那么对于一个集合的石头，最后最少可以保留一个石头。其他石头一定可以被清除掉。

Q959：由斜杠划分区域 一个方框划分为 0,1,2,3 上右下左四个区域。方块内如果是'那么01，23是相连的。 '/'同理。

1. 方块之间的相连情况是：1连向右边一个方块的3. 2连向下边一个方块的0. 这样把所有方块内，方块间的关系连在一起。就完成了并查集的合并。

Q1202：交换字符串中的元素: $\text{pairs}[i] = [a, b]$ 表示字符串中的两个索引. 可以任意多次交换 在 pairs 中任意一对索引处的字符

0. 返回在经过若干次交换后， s 可以变成的按字典序最小的字符串。
1. 我们这么想，若 a 属于集合 $\text{set}(a)$, b 属于集合 $\text{set}(b)$ ，那么 ab 就会将两个集合合并在一起，要一起参与排序。所以恰好是并查集。
2. 关于排序: 使用 $\text{int}[26]$ 对每个集合分配一个 $\text{int}[26]$ s
3. 将集合中的所有下标，统计其字符频率，那么最后的顺序就是 $\text{int}[26]$ 的顺序组合。

```
int[][] map = new int[lens][26];
for(int i = 0; i < lens; i++){
    int root = find(i);
    map[root][s.charAt(i) - 'a']++;
}
```

拼接: ansStr 我们通过下标 i 确定 ansStr 的每一位应该用什么字符。如果 i 的跟是 root 。那么在 root 的字符频率表中即 $\text{int[]} \text{freq} = \text{map}[\text{root}]$ 在 freq 中找第一个字符频率不为 0 的字符。就是位置 i 应该设置的字符。然后对频率--。

并查集的使用说明

1. 百分之80的图论题，可以用并查集来解决。
2. 并查集是集合问题，题意中一般都会涉及到，划分集合的动作。如牵手，可达，并集，相似，等等。
3. 二维图论中并查集的使用，二维图中一般都是双坐标 (x, y) 。我们一般会做一个坐标映射， $\text{getindex}(x, y) = \text{index}$. 用来表示该点在并查集数组中的下标。二维到一维的映射。